IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of: Dettinger et al.	§	Confirmation No.: 1355
	§	
	§	Group Art Unit: 2165
Serial No.: 10/733,973	§	
	§	Examiner: Tomasz Ponikiewski
Filed: December 11, 2003	§	
	§	
For: REUSING INTERMEDIATE WORKFLOW RESULTS IN SUCCESSIVE WORKFLOW RUNS	§	

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief - Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450, or facsimile transmitted to the U.S. Patent and Trademark Office to fax number 571-273-8300 to the attention of Examiner Tomasz Ponikiewski, or electronically transmitted via EFS-Web, on the date shown below:

June 14, 2007 /Esther Marques/
Date Esther Marques

Dear Sir:

APPEAL BRIEF

Applicants submit this Appeal Brief to the Board of Patent Appeals and Interferences on appeal from the decision of the Examiner of Group Art Unit 2165 dated November 15, 2006, finally rejecting claims 1-12, 15-30 and 32-35. The final rejection of claims 1-12, 15-30 and 32-35 is appealed. This Appeal Brief is believed to be timely since it is electronically transmitted by the extended due date of June 15, 2007, as set by the filing of a Notice of Appeal on March 15, 2007. Please charge the fee of $500.00 for filing this brief to Deposit Account No. 09-0465/ROC920030307US1.

TABLE OF CONTENTS

Real Party in Interest

The present application has been assigned to International Business Machines Corporation, Armonk, New York.

Related Appeals and Interferences

Applicants assert that no other appeals or interferences are known to the Applicants, the Applicants legal representative, or assignee which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

Status of Claims

Claims 1-12, 15-30 and 32-35 are pending in the application. Claims 1-34 were originally presented in the application. Claim 35 has been added during prosecution. Claims 13-14 and 31 have been canceled without prejudice. Claims 1-12, 15-30 and 32-35 stand finally rejected as discussed below. The final rejections of claims 1-12, 15-30 and 32-35 are appealed. The pending claims are shown in the attached Claims Appendix.

Status of Amendments

All claim amendments have been entered by the Examiner, including amendments to the claims proposed after the final rejection.

Summary of Claimed Subject Matter

A. CLAIM 1 – INDEPENDENT

Claim 1 recites a computer-implemented method of executing a multi-step workflow that is repeatedly executed on relevant data of a database. See *Application*, page 3: lines 31-32; page 11: lines 7-9; page 20: lines 3-5; page 23: lines 25-26; Figure 1, 146, 156; Figure 3A (shows one execution runtime, as described on page 20: lines 7-9.). The workflow consists of a plurality of steps, where each step is an executable function that takes input from a previous step and produces output for a subsequent step. See *Application*, 4: 15-19; 11: 9-11 (see 11: 14-17 for definition of functional module); 15: 16-18; Figure 2, 162, 165, 222. During execution a step receives current input from the preceding function, and in the previous execution runtime the step received identical input from the preceding function and produced output while operating on the relevant data of the database. See *Application*, 3: 32-34, 4:1; 7: 5-12; 15: 16-22; 21: 17-19, 29-32; Figure 4, 306A, 309A, 404.

As claimed, the method includes determining whether the step is deterministic, *i.e.*, given the same input and the same relevant data, the step will always generate the same output. See *Application*, 4: 1; 21: 19-23; Table II: 012; Figure 4, 402. If the step is deterministic, do not re-execute the step; instead, return the output produced during the previous execution of the step. See *Application*, 4: 3-4; 7: 10-12; 13:-14-17; 22: 1-4; Figure 4, 308, 406.

B. CLAIM 10 - INDEPENDENT

Claim 10 recites a computer-implemented method of managing the execution of a multi-step workflow that is repeatedly executed on relevant data of a database. See *Application*, 3: 31-32; 11: 7-9; 20: 3-5; 23: 25-26; Figure 1, 146, 156; Figure 3A (shows one execution runtime, as described on 20: 7-9.). The workflow consists of a plurality of steps, where each step is an executable function that takes input from a previous step and produces output for a subsequent step. See *Application*, 4: 15-19; 11: 9-11 (see

11: 14-17 for definition of functional module); 15: 16-18; Figure 2, 162, 165, 222. During execution a step receives current input from the preceding function, and in the previous execution runtime the step received identical input from the preceding function and produced output while operating on the relevant data of the database. See *Application*, 3: 32-34, 4:1; 7: 5-12; 15: 16-22; 21: 17-19, 29-32; Figure 4, 306A, 309A, 404.

As claimed, the method includes identifying whether the step is deterministic, *i.e.*, given the same input and the same relevant data, the step will always generate the same output. See *Application*, 4: 1; 21: 19-23; Table II: 012; Figure 4, 402. If the step has been previously executed using input identical to the current input, the method does not re-execute the step and instead returns the output previously produced. See *Application*, 4: 3-4; 7: 10-12; 13:-14-17; 22: 1-4; Figure 4, 308, 406. If the step has not been previously executed using input identical to the current input, execute the step and store the result to enable future skipping of this execution step, as described in the previous sentence, 21: 32-36, 22: 1; Figure 3A, 307, 308, 309A; Figure 4, 307, 308, 309A, 404, 410.

C. CLAIM 15 - INDEPENDENT

Claim 15 recites computer readable storage medium containing a program which, when executed by a processor, manages the execution of a multi-step workflow that is repeatedly executed on relevant data of a database. See *Application*, 4: 13-16; 11: 7-9; 20: 3-5; 23: 25-26; Figure 1, 132, 146, 156; Figure 3A (shows one execution runtime, as described on page 20: lines 7-9.). The workflow consists of a plurality of steps, where each step is an executable function that takes input from a previous step and produces output for a subsequent step. See *Application*, 4: 15-19; 11: 9-11 (see 11: 14-17 for definition of functional module); 15: 16-18; Figure 2, 162, 165, 222. During execution a step receives current input from the preceding function, and in the previous execution runtime the step received identical input from the preceding function and produced output while operating on the relevant data of the database. See *Application*, 4: 14-19; 7: 5-12; 15: 16-22; 21: 17-19, 29-32; Figure 4, 306A, 309A, 404.

As claimed, management of the workflow by the program includes determining whether the step is deterministic, *i.e.*, given the same input and the same relevant data, the step will always generate the same output. See *Application*, 4: 20; 21: 19-23; Table II: 012; Figure 4, 402. If the step is deterministic, do not re-execute the step; instead, return the output produced during the previous execution of the step. See *Application*, 4: 21-23; 7: 10-12; 13:-14-17; 22: 1-4; Figure 4, 308, 406.

D. CLAIM 24 - INDEPENDENT

Claim 24 recites computer readable storage medium containing a program which, when executed by a processor, manages the execution of a multi-step workflow that is repeatedly executed on relevant data of a database. See *Application*, 4: 13-16; 11: 7-9; 20: 3-5; 23: 25-26; Figure 1, 132, 146, 156; Figure 3A (shows one execution runtime, as described on page 20: lines 7-9.). The workflow consists of a plurality of steps, where each step is an executable function that takes input from a previous step and produces output for a subsequent step. See *Application*, 4: 15-19; 11: 9-11 (see 11: 14-17 for definition of functional module); 15: 16-18; Figure 2, 162, 165, 222.

As claimed, during execution of the program a step receives current input from the preceding function, and in the previous execution runtime the step received identical input from the preceding function and produced output while operating on the relevant data of the database. See *Application*, 4: 14-19; 7: 5-12; 15: 16-22; 21: 17-19, 29-32; Figure 4, 306A, 309A, 404. The current step generates identical output for given input in repeated executions of the step on the relevant data; the output produced during the previous execution of the step is returned without re-executing the current step. See *Application*, 4: 20-23; 7: 10-12; 13:-14-17; 22: 1-8; Figure 4, 308, 402, 406.

E. CLAIM 29 - INDEPENDENT

Claim 29 recites a computer system including a database with data and a workflow execution manager residing in memory for managing execution of a multi-step workflow that is repeatedly executed on the data of the database. See *Application*, 5: 1-4; 10: 17-19; 11: 7-9; 20: 3-5; 23: 25-26; Figure 1, 132, 146, 156; Figure 3A (shows one

execution runtime, as described on page 20: lines 7-9.). The workflow consists of a plurality of steps, where each step is an executable function that takes input from a previous step and produces output for a subsequent step. See *Application*, 5: 6-11; 11: 9-11 (see 11: 14-17 for definition of functional module); 15: 16-18; Figure 2, 162, 165, 222. The workflow execution manager is configured to receive current input to a step of the workflow from the preceding function, and in the step has previously received identical input from the preceding function and produced output while executing on the relevant data of the database. See *Application*, 5: 4-8; 7: 5-12; 15: 16-22; 21: 17-19, 29-32; Figure 4, 306A, 309A, 404.

As claimed, management of the workflow by the program includes determining whether the step is deterministic, *i.e.*, given the same input and the same relevant data, the step will always generate the same output. See *Application*, 5: 8; 21: 19-23; Table II: 012; Figure 4, 402. If the step is deterministic, do not re-execute the step; instead, return the output produced during the previous execution of the step. See *Application*, 5: 10-12; 7: 10-12; 13:-14-17; 22: 1-4; Figure 4, 308, 406.

F. CLAIM 30 - INDEPENDENT

Claim 30 recites a computer system including a database with data and a workflow execution manager residing in memory for managing execution of a multi-step workflow that is repeatedly executed on the data of the database. See *Application*, 5: 1-4; 10: 17-19; 11: 7-9; 20: 3-5; 23: 25-26; Figure 1, 132, 146, 156; Figure 3A (shows one execution runtime, as described on page 20: lines 7-9.). The workflow consists of a plurality of steps, where each step is an executable function that takes input from a previous step and produces output for a subsequent step. See *Application*, 5: 6-11; 11: 9-11 (see 11: 14-17 for definition of functional module); 15: 16-18; Figure 2, 162, 165, 222.

As claimed, the workflow execution manager is configured to receive current input to a step of the workflow from the preceding function, and the step has previously received identical input from the preceding function and produced output while executing on the relevant data of the database. See *Application*, 5: 4-8; 7: 5-12; 15: 16-

22; 21: 17-19, 29-32; Figure 4, 306A, 309A, 404. The current step generates identical output for given input in repeated executions of the step on the relevant data; the output produced during the previous execution of the step is returned without re-executing the current step. See *Application*, 7: 10-12; 13:-14-17; 22: 1-8; Figure 4, 308, 402, 406.

G. CLAIM 32 - INDEPENDENT

Claim 32 recites a computer-implemented method of automatically executing a plurality of functional modules from within an application. See *Application,* 14: 25-27; 16: 26-28; 18: 25-26; 23: 18-22; Figure 2, 161, 162, 210; Figure 3B, 304 and 305. As claimed, the method provides an interface for specifying a single multi-analysis functional module used to execute the plurality of functional modules, whereby user selection of the single multi-analysis functional module is an implicit selection of the plurality of functional modules. See *Application*, 12: 30-33, 13:1-5; Figure 2, 161 and 210. Each of the plurality of functional modules is an executable function that takes input from a previous step and produces output for a subsequent step. See *Application*, 11: 9-11 (see 11: 14-17 for definition of functional module); 15: 16-18; Figure 2, 162, 165, 222.

The method is configured for the current functional module(s) to receive current input from the preceding functional module(s), and in the previous execution runtime the current module(s) received identical input from the preceding module(s) and produced output. See *Application*, 7: 5-12; 15: 16-22; 21: 17-19, 29-32; Figure 4, 306A, 309A, 404. As claimed, the method also includes determining whether at least one functional module is deterministic, in that the at least one functional module generates identical output for given input in repeated executions of the at least one functional module. See *Application*, 21: 19-23; Table II: 012; Figure 4, 402. Note that multi-analysis modules may simply invoke the same determinism identifying mechanism used by single-analysis modules. See *Application*, 20: 15-17.

If at least one functional module is deterministic, then for the deterministic module(s) do not re-execute and, instead, return the output produced during the

previous execution of the step. See *Application*, 5: 10-12; 7: 10-12; 13:-14-17; 22: 1-4; Figure 4, 308, 406.

H. CLAIM 35 - INDEPENDENT

Claim 35 recites a computer-implemented method of managing the execution of a multi-step workflow that is repeatedly executed on relevant data of a database. See *Application*, 11: 7-9; 20: 3-5; 23: 25-26; Figure 1, 146, 156; Figure 3A (shows one execution runtime, as described on 20: 7-9.). The workflow consists of a plurality of steps, where each step is an executable function that takes input from a previous step and produces output for a subsequent step. See *Application*, 11: 9-11 (see 11: 14-17 for definition of functional module); 15: 16-18; Figure 2, 162, 165, 222. During execution a step receives current input from the preceding function, and in the previous execution runtime the step received identical input from the preceding function and produced output while operating on the relevant data of the database. See *Application*, 7: 5-12; 15: 16-22; 21: 17-19, 29-32; Figure 4, 306A, 309A, 404.

As claimed, the method includes identifying whether the step is deterministic, *i.e.*, given the same input and the same relevant data, the step will always generate the same output. See *Application*, 4: 1; 21: 19-23; Table II: 012; Figure 4, 402. Upon determining that the step has been previously executed using input identical to the current input, determining whether the relevant data has been changed since the previous execution of the step using the input identical to the current input. See *Application*, 22: 18-29; Figure 4, 404.

As claimed, if the relevant data has been changed since the previous execution of the step, output will not be identical, so the method will re-execute the step and store the result. See Application, 21: 32-36, 22: 1; Figure 3A, 307, 308, 309A; Figure 4, 307, 308, 309A, 404, 410. If the relevant data has not been changed since the previous execution of the step, then the method does not re-execute the step and instead returns the output previously produced. See *Application*, 7: 10-12; 13:-14-17; 22: 1-4, 26-29; Figure 4, 308, 406.

Grounds of Rejection to be Reviewed on Appeal

1. Rejection of claims 1-12, 15-28 and 35 under 35 U.S.C. 102(e) as being anticipated by *Li* (U.S. Patent No. 6,748,386).

2. Rejection of claims 29-30 and 32-34 under 35 U.S.C. 103(a) as being unpatentable over *Li* in view of *Crisan et al*. (U.S. Publication 2003/0191769, hereinafter, "*Crisan*").

ARGUMENTS

Rejection of claims 1-12, 15-28 and 35 under 35 U.S.C. 102(e) as being anticipated by *Li*.

"A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as is contained in the ... claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). The elements must be arranged as required by the claim. *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990).

Claims 1, 10, 15 and 35 Are Not Anticipated under 35 U.S.C. § 102(e) by *Li*

Respectfully, Applicants submit that *Li* does not disclose a computer-implemented method of (i) execution of a multi-step workflow that is repeatedly executed on data of a database, (ii) that defines workflow as a plurality of steps, each step being an executable function that operates on input from a previous step and produces output for a subsequent step and each step having a defined input format and output format, (iii) receiving current input to a step of the workflow on relevant data of the database, (iv) wherein the step has been previously executed on the relevant data using previous input identical to the current input and wherein the previous execution of the step produced previous output, (v) determining whether the step is deterministic (vi) if the step is deterministic, returning the previous output produced during the previous execution of the step without re-executing the step.

The Examiner, citing to *Li* at column 6, lines 55-56 and column 7, lines 55-61, states that *Li* teaches "receiving current input for execution of a step of the workflow on relevant data of the database, wherein the step has been previously executed on the

relevant data using previous identical input to the current input and wherein the previous execution of the step produced previous output."

As an initial matter, *Li* is not directed to "workflow." Workflow is a well-defined term in which a plurality of components, or modules, cooperatively performs predefined operations to provide a result. Each workflow component performs a predefined transformative operation (step) on some input and produces corresponding output. The workflow defines a particular order in which the components are invoked, and the output of one workflow component is provided as input to another workflow component. In this way, independent functional modules having defined interfaces cooperate with one another (on an input/output basis) to achieve a result. This definition of workflow is well-known. *See, e.g.*, Wikipedia (keyword "workflow"); *see also,* present application at, *e.g.*, paragraphs 0060-0062.

In contrast, *Li* teaches a method for "query mappings in which the users do not need to manually specify such mappings." *See* col. 3, lines 10-12. This is not transformative as understood in the context of workflows. In addition, the input taught by *Li* (queries 68 or requests 72) referred to by the Examiner does not invoke a workflow. Rather, the queries 68 and requests 72 are merely requests to information directed to different sources, *i.e.* the DBMS 76 and the file system and network 74, respectively. See, *Li* at col. 7, lines 57-61. Accordingly, withdrawal of this rejection is respectfully requested.

Furthermore, *Li* does not teach "determining whether the step is deterministic, whereby the step generates identical output for given input in repeated executions of the step on the relevant data, and, if the step is deterministic, returning the previous output produced during the previous execution of the step without re-executing the step." In this regard, the Examiner cites column 7, lines 54-61 and argues that *Li* "the cached results would not be retrieved if the query wasn't the same." Respectfully, the Examiner's argument misconstrues the definition of a deterministic step. The issue is not whether the query is the same but, rather, given the same query, whether the same output is always produced. This is the fundamental characteristic of a deterministic

step, as widely understood in the art. *See, e.g.*, Wikipedia (keyword "deterministic algorithm"). Caching, as taught by *Li*, instead considers whether, given the same query, the previously generated output is still a valid result to return. This is not strictly equivalent to whether the output is the same. For example, the method disclosed by *Li* could cache the result of a non-deterministic step (*i.e.*, output varies over identical input) whose solution has been approximated using a Monte Carlo based function. *See, e.g.*, Wikipedia (keyword ("Monte Carlo method"). In such a case, for each repeated execution of the step over the same input and relevant data the output will not be identical, but instead fall within some defined acceptable range. The cache, as taught by *Li*, can be configured to save a representative example of the varying results and simply return the representative example every time in lieu of re-executing the step. Thus, even if the step does not generate identical output for given input in repeated executions of the step on the relevant data, caching as disclosed in *Li* could still save, and later return, the output. Hence, *Li* does not teach either determining whether a step is deterministic or leveraging that knowledge to return the previously generated identical output.

In addition, the Examiner asserts in the advisory action that "[t]he limitation does not claim that an execution of a step is made to determine if the step is deterministic. The examiner therefore broadly interprets the *Li* reference to read on the limitation listed in the current application." This argument is untenable, as determining whether a step is deterministic is sufficient to distinguish the claimed invention from *Li*, for the reasons outlined above. How this determining is done is irrelevant to the distinction. Accordingly, withdrawal of this rejection is respectfully requested.

Claims 10, 15 and 35 have similar limitations and therefore are allowable for all or some of the reasons given above.

Therefore, the claims are believed to be allowable, and allowance of the claims is respectfully requested.

Claims 2-9 Are Not Anticipated under 35 U.S.C. § 102(e) by *Li*

Claims 2-9 depend from independent claim 1, and are therefore believed to be allowable for the reasons provided above. Accordingly, withdrawal of this rejection is respectfully requested.

Claims 11-12 Are Not Anticipated under 35 U.S.C. § 102(e) by *Li*

Claims 11-12 depend from independent claim 10, and are therefore believed to be allowable for the reasons provided above. Accordingly, withdrawal of this rejection is respectfully requested.

Claims 16-23 Are Not Anticipated under 35 U.S.C. § 102(e) by *Li*

Claims 16-23 depend from independent claim 15, and are therefore believed to be allowable for the reasons provided above. Accordingly, withdrawal of this rejection is respectfully requested.

Claim 24 Is Not Anticipated under 35 U.S.C. § 102(e) by *Li*

Respectfully, Applicants submit that *Li* does not read upon the claimed invention for the same reasons as described in the previous section, except for the analysis regarding deterministic steps. Instead, while claim 24 does not determine whether a step is deterministic, it does claim "the step generates identical output for given input in repeated executions of the step on the relevant data."

As detailed in the deterministic step analysis above, caching, as taught by *Li*, does not require that a step generate identical output for given input in repeated executions of the step. Caching instead considers whether, given the same query, the previously generated output is still a valid result to return. This is not the same as identical output. For example, the method disclosed by *Li* could cache the result of a step where the output generated in response to the same query varies, but falls within some defined acceptable range. Thus, even if the step does not generate identical output for given input in repeated executions of the step on the relevant data, caching as disclosed in *Li* could still save and later return the non-identical output. Hence, *Li* does not teach that the step generates identical output for given input in repeated

executions of the step on the relevant data. Accordingly, withdrawal of this rejection is respectfully requested.

Claims 25-28 Are Not Anticipated under 35 U.S.C. § 102(e) by *Li*

Claims 25-28 depend from independent claim 24, and are therefore believed to be allowable for the reasons provided above. Accordingly, withdrawal of this rejection is respectfully requested.

2. Rejection of claims 29-30 and 32-34 under 35 U.S.C. 103(a) as being unpatentable over *Li* in view of *Crisan*.

The Examiner bears the initial burden of establishing a *prima facie* case of obviousness. *See* MPEP § 2142. To establish a *prima facie* case of obviousness three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one ordinary skill in the art to modify the reference or to combine the reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *See* MPEP § 2143. The present rejection fails to establish at least the third criterion.

Claims 29-30 and 32 Are Not Obvious under 35 U.S.C. § 103(a) over *Li* in view of *Crisan*

Specifically, *Li* does not teach, show or suggest "receiving current input to a step of the workflow on relevant data of the database, wherein the step has been previously executed on the relevant data using previous input identical to the current input and wherein the previous execution of the step produced previous output" as established above with respect to the 102(e) rejection. Further, *Li* does not teach, show or suggest "determining whether the step is deterministic, in that the step generates identical output for given input in repeated executions of the step on the relevant data, as established above. Therefore, the combination of *Li* and *Crisan* does not disclose each of the claimed elements of the rejected claims. Accordingly, withdrawal of this rejection is respectfully requested. Claims 30 and 32 recite similar limitations.

Claims 29-30 Are Not Obvious under 35 U.S.C. § 103(a) over *Li* in view of *Crisan*

Examiner asserts that "[i]t would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the *Li* teachings of *Crisan* [sic] to include providing an interface for specifying a single multi-analysis functional module used to execute the plurality of functional modules, wherein the workflow is defined by plurality of steps, each step being an executable function that operates on input from a previous step and produces output for a subsequent step and each having a defined input format and output format because an output of one operation could be useful as an input for another." Respectfully, Applicants note that claims 29 and 30 do not recite a multi-analysis functional module. Thus, even assuming, *arguendo*, that the combination of *Li* and *Crisan* teaches a multi-analysis functional module, such a teaching is irrelevant. Therefore the Examiner's rejection is defective and withdrawal of this rejection is respectfully requested.

Claim 32 Is Not Obvious under 35 U.S.C. § 103(a) over *Li* in view of *Crisan*

The Examiner asserts that *Crisan* teaches "providing an interface for specifying a single multi-analysis functional module used to execute the plurality of functional modules, wherein the workflow is defined by plurality of steps, each step being an executable function that operates on input from a previous step and produces output for a subsequent step and each having an defined input format and output format" at paragraph 0130, lines 4-15. Respectfully, Applicants note that the cited lines only define a workflow, nothing more. There is no mention of a multi-analysis functional module; even with a broad reading of the passage, *Crisan* does not disclose a multi-analysis functional module that, as defined in the claim, is used to execute a plurality of functional modules. The multi-analysis functional module is a process invoked as part of a workflow, not the workflow itself. Furthermore, *Crisan* is directed towards a means of assembling a workflow across multiple computing platforms, not multi-analysis functional modules. See page 1, paragraph 0006 ("There is a need in the art to develop techniques to utilize workflow technologies with different computing platforms."). Thus, *Crisan* does not teach the claimed element.

Accordingly, withdrawal of this rejection is respectfully requested.

Claims 33 and 34 Are Not Obvious under 35 U.S.C. § 103(a) over *Li* in view of *Crisan*

Claims 33 and 34 depend from independent claim 32, and are therefore believed to be allowable for the reasons provided above. Accordingly, withdrawal of this rejection is respectfully requested.

CONCLUSION

The Examiner errs in finding that:

1. Claims 1-12, 15-28 and 35 are anticipated by *Li*; and

2. Claims 29-30 and 32-34 are unpatentable over *Li* in view of *Crisan*.

Withdrawal of the rejections and allowance of all claims is respectfully requested.

Respectfully submitted, and
S-signed pursuant to 37 CFR 1.4,

/Gero G. McClellan, Reg. No. 44,227/

Gero G. McClellan
Registration No. 44,227
Patterson & Sheridan, L.L.P.
3040 Post Oak Blvd. Suite 1500
Houston, TX 77056
Telephone: (713) 623-4844
Facsimile: (713) 623-4846
Attorney for Appellants

CLAIMS APPENDIX

1. (Previously Presented) A computer-implemented method of execution of a multi-step workflow that is repeatedly executed on data of a database, wherein the workflow is defined by a plurality of steps, each step being an executable function that operates on input from a previous step and produces output for a subsequent step and each step having a defined input format and output format, the method comprising:

receiving current input to a step of the workflow on relevant data of the database, wherein the step has been previously executed on the relevant data using previous input identical to the current input and wherein the previous execution of the step produced previous output;

determining whether the step is deterministic, in that the step generates identical output for given input in repeated executions of the step on the relevant data; and

if the step is deterministic, returning the previous output produced during the previous execution of the step without re-executing the step.

2. (Original) The method of claim 1, further comprising using the returned previous output as input to a next sequential step in the workflow.

3. (Original) The method of claim 1, wherein determining whether the step is deterministic comprises determining whether a workflow description of the step includes a deterministic flag indicating that the step generates identical output for given input in repeated executions of the step on the relevant data.

4. (Original) The method of claim 1, further comprising:

determining whether the current input and the previous input are the same; and

returning the previous output produced during the previous execution of the step only if the current input and the previous input are determined to be the same.

5. (Original) The method of claim 4, wherein determining whether the current input and the previous input are the same comprises accessing a hash table representative of the previous input.

6. (Original) The method of claim 1, further comprising:

determining whether the relevant data has been changed since the previous execution; and

returning the previous output produced during the previous execution of the step only if the relevant data has not been changed.

7. (Original) The method of claim 6, wherein determining whether the relevant data has been changed comprises:

determining a timestamp indicating a point of time of the previous execution; and

determining, from a transaction log of the database, whether transactions relative to the relevant data have occurred since the point of time indicated by the timestamp.

8. (Previously Presented) The method of claim 6, further comprising:

if the relevant data has been changed since the previous execution:

executing the step on the relevant data to obtain a result; and

storing the result as output to be returned for subsequent invocations of the step taking input identical to the current input, in which case execution of the step is avoided and the stored output is returned for the step.

9. (Original) The method of claim 1, wherein the current input comprises one or more result fields and input parameters.

10. (Previously Presented) A computer-implemented method of managing execution of a workflow that is repeatedly executed on data of a database, wherein the workflow is defined by a plurality of steps, each step being an executable function that operates on input from a previous step and produces output for a subsequent step and each step having a defined input format and output format, the method comprising:

receiving current input for execution of a step of the workflow on relevant data of the database;

identifying the step as deterministic, in that the step generates identical output for given input in repeated executions of the step on the relevant data;

upon determining that the step has been previously executed using input identical to the current input:

returning output obtained in the previous execution of the step using input identical to the current input without executing the step using the received current input; and

upon determining that the step has not been previously executed using input identical to the current input:

executing the step for the current input on the relevant data to obtain a result; and

storing the result to enable managing a next invocation of the step in which the step is passed input identical to the current input, in which case the stored result is returned as output for the step without re-executing the step.

11. (Previously Presented) The method of claim 10, further comprising, upon determining that the step has been previously executed using input identical to the current input and prior to returning the output:

determining whether the relevant data has been changed since the previous execution of the step using the input identical to the current input; and

if the relevant data has not been changed, retrieving the output obtained in the previous execution of the step using the input identical to the current input.

12. (Original) The method of claim 11, wherein determining whether the relevant data has been changed comprises:

determining a timestamp indicating a point of time of the previous execution; and

determining, from a transaction log of the database, whether transactions relative to the relevant data have occurred since the point of time indicated by the timestamp.

13-14. (Canceled)

15. (Previously Presented) A computer readable storage medium containing a program which, when executed by a processor, performs an operation of managing execution of a multi-step workflow that is repeatedly executed on data of a database, wherein the workflow is defined by a plurality of steps, each step being an executable function that operates on input from a previous step and produces output for a

subsequent step and each step having a defined input format and output format, the operation comprising:

receiving current input to a step of the workflow on relevant data of the database, wherein the step has been previously executed on the relevant data using previous input identical to the current input and wherein the previous execution of the step produced previous output;

determining whether the step is deterministic, in that the step generates identical output for given input in repeated executions of the step on the relevant data; and

if the step is deterministic, returning the previous output produced during the previous execution of the step without re-executing the step.

16. (Previously Presented) The computer readable storage medium of claim 15, wherein the operation further comprises:

inputting the returned previous output to a next sequential step in the workflow.

17. (Previously Presented) The computer readable storage medium of claim 15, wherein determining whether the step is deterministic comprises determining whether a workflow description of the step includes a deterministic flag indicating that the step generates identical output for given input in repeated executions of the step on the relevant data.

18. (Previously Presented) The computer readable storage medium of claim 15, wherein the operation further comprises:

determining whether the current input and the previous input are the same; and

returning the previous output produced during the previous execution of the step only if the current input and the previous input are determined to be the same.

19. (Previously Presented) The computer readable storage medium of claim 18, wherein determining whether the current input and the previous input are the same comprises accessing a hash table representative of the previous input.

20. (Previously Presented) The computer readable storage medium of claim 15, wherein the operation further comprises:

determining whether the relevant data has been changed since the previous execution; and

returning the previous output produced during the previous execution of the step only if the relevant data has not been changed.

21. (Previously Presented) The computer readable storage medium of claim 20, wherein determining whether the relevant data has been changed comprises:

retrieving a timestamp indicating a point of time of the previous execution; and

retrieving a transaction log of the database; and

determining, from the transaction log, whether transactions relative to the relevant data have occurred since the point of time indicated by the timestamp.

22. (Previously Presented) The computer readable storage medium of claim 20, wherein the operation further comprises:

if the relevant data has been changed since the previous execution:

executing the step on the relevant data to obtain a result; and

storing the result as output to be returned for subsequent invocations of the step taking input identical to the current input, in which case execution of the step is avoided and the stored output is returned for the step.

23. (Previously Presented) The computer readable storage medium of claim 15, wherein the current input comprises one or more result fields and input parameters.

24. (Previously Presented) A computer readable storage medium containing a program which, when executed by a processor, performs an operation of managing execution of a workflow that is repeatedly executed on data of a database, wherein the workflow is defined by a plurality of steps, each step being an executable function that operates on input from a previous step and produces output for a subsequent step and each step having a defined input format and output format, the operation comprising:

receiving current input for execution of a step of the workflow on relevant data of the database, wherein the step generates identical output for given input in repeated executions of the step on the relevant data; and

without executing the step using the current input, returning output obtained in a previous execution of the step using input identical to the current input.

25. (Previously Presented) The computer readable storage medium of claim 24, wherein the operation further comprises, prior to returning the output:

determining whether the step has been previously executed using the input identical to the current input;

if so, determining whether the relevant data has been changed since the previous execution of the step using the input identical to the current input; and

if the relevant data has not been changed, retrieving the output obtained in the previous execution of the step using the input identical to the current input.

26. (Previously Presented) The computer readable storage medium of claim 25, wherein determining whether the relevant data has been changed comprises:

retrieving a timestamp indicating a point of time of the previous execution;

retrieving a transaction log of the database; and

determining, from the transaction log, whether transactions relative to the relevant data have occurred since the point of time indicated by the timestamp.

27. (Previously Presented) The computer readable storage medium of claim 25, wherein the operation further comprises:

if the step has not been executed using the input identical to the current input:

executing the step for the current input on the relevant data to obtain a result; and

storing the result to enable managing a next invocation of the step in which the step is passed input identical to the current input, in which case the stored result is returned as output for the step without re-executing the step.

28. (Previously Presented) The computer readable storage medium of claim 25, wherein the operation further comprises:

if the relevant data has been changed since the previous execution of the step using the input identical to the current input:

executing the step for the current input on the relevant data to obtain a result; and

storing the result to enable managing a next invocation of the step in which the step is passed input identical to the current input, in which case the stored result is returned as output for the step without re-executing the step.

29. (Previously Presented) A computer system, comprising:

a database having data; and

a workflow execution manager residing in memory for managing execution of a multi-step workflow that is repeatedly executed on the data of the database, wherein the workflow is defined by a plurality of steps, each step being an executable function that operates on input from a previous step and produces output for a subsequent step and each step having a defined input format and output format, the workflow execution manager being configured for:

receiving current input to a step of the workflow on relevant data of the database, wherein the step has been previously executed on the relevant data using previous input identical to the current input and wherein the previous execution of the step produced previous output;

determining whether the step is deterministic, in that the step generates identical output for given input in repeated executions of the step on the relevant data; and

if the step is deterministic, returning the previous output produced during the previous execution of the step without re-executing the step.

30. (Previously Presented) A computer system, comprising:

a database having data; and

a workflow execution manager residing in memory for managing execution of a workflow that is repeatedly executed on the data of the database, wherein the workflow is defined by a plurality of steps, each step being an executable function that operates on input from a previous step and produces output for a subsequent step and each step having a defined input format and output format, the workflow execution manager being configured for:

receiving current input to a step of the workflow on relevant data of the database, wherein the step generates identical output for given input in repeated executions of the step on the relevant data; and

without executing the step using the current input, returning output obtained in a previous execution of the step using input identical to the current input.

31. (Canceled)

32. (Previously Presented) A computer-implemented method of automatically executing a plurality of functional modules from within an application, comprising:

providing an interface for specifying a single multi-analysis functional module used to execute the plurality of functional modules, whereby user selection of the single multi-analysis functional module is an implicit selection of the plurality of functional modules, and wherein each of the plurality of functional modules is an executable function that operates on input from a previous functional module and produces output for a subsequent functional module and each functional module having a defined input format and output format;

receiving current input to at least one of the functional modules, wherein the at least one functional module has been previously executed using previous input identical to the current input;

determining whether the at least one functional module is deterministic, in that the at least one functional module generates identical output for given input in repeated executions of the at least one functional module; and

if the at least one functional module is deterministic, returning previous output produced during the previous execution without re-executing the at least one functional module.

33. (Original) The method of claim 32, further comprising retrieving information regarding execution of the plurality of functional modules from a configuration file.

34. (Original) The method of claim 33, wherein determining whether the at least one functional module is deterministic comprises examining information regarding the at least one functional module retrieved from the configuration file.

35. (Previously Presented) A computer-implemented method of managing execution of a workflow that is repeatedly executed on data of a database, wherein the workflow is defined by a plurality of steps, each step being an executable function that operates on input from a previous step and produces output for a subsequent step and each step having a defined input format and output format, the method comprising:

receiving current input for execution of a step of the workflow on relevant data of the database;

identifying the step as deterministic, in that the step generates identical output for given input in repeated executions of the step on the relevant data;

upon determining that the step has been previously executed using input identical to the current input, determining whether the relevant data has been changed since the previous execution of the step using the input identical to the current input;

if the relevant data has been changed since the previous execution of the step using the input identical to the current input:

executing the step for the current input on the relevant data to obtain a result; and

storing the result to enable managing a next invocation of the step in which the step is passed input identical to the current input, in which case the stored result is returned as output for the step without re-executing the step; and

if the relevant data has not been changed since the previous execution of the step using the input identical to the current input:

returning output obtained in the previous execution of the step using the input identical to the current input without executing the step using the received current input.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.